ChatGPT

SON GitHub Implementation Blueprint for a Promptable Image-Segmentation Service

Connectors enabled (and consulted): Lovable [1], GitHub [2], Replit [3].

Executive summary

This report specifies a concrete, production-style ("Cisco-style": clean structure, explicit governance, repeatable CI/CD, compliance-first data handling) implementation plan for a **public** repository ("SON GitHub") that hosts a promptable image-segmentation service with (a) a Dockerised REST API and (b) a small CLI client. The emphasis is on: (1) **public redistribution safety** (no raw datasets, no ambiguous/proprietary weights in git), (2) **repeatability** (pin dependencies, deterministic modes, container pinning), and (3) **operational maturity** (CI, releases, monitoring hooks, security policies). Guidance is grounded in official documentation for GitHub features (Actions, Pages, Releases, branch protection) and official dataset homepages (Open Images, DAVIS, YouTube-VOS, COCO, VOC). [4]

Key conservative design choices (recommended for a public demo repo):

- **Model artefacts are never committed to git**; instead they are fetched at runtime and verified via checksum. This avoids GitHub's hard file limits (100 MiB blocks) and keeps the repo small and forkable. [5]
- **Model artefacts are distributed via GitHub Releases by default**, because Releases support large binaries without repo bloat (up to 2 GiB per asset; up to 1000 assets per release; no total size/bandwidth limits). [6]
- **Datasets are consumed only via download scripts**; the repo contains dataset "cards" and attribution files, and validation scripts, but never stores raw images. This is essential because several popular segmentation datasets have licence constraints or provenance requirements (e.g., VOC uses Flickr images; YouTube-VOS is non-commercial research only). [7]
- **CI workflows are pinned, least-privilege, and supply-chain aware** (pin Actions to SHAs as recommended; use minimal token permissions per job; enable secret scanning, dependency updates, and CodeQL scanning). [8]

Connectors used

- GitHub connector: used to locate and read the selected repository `Zoro7781/Projects` (found, public, minimal current content). No other GitHub repositories were used as evidence to comply with the constraint "do not use other GitHub repos."
- Lovable connector: available, but requires an explicit `project_id` to inspect a build; none was provided, so no Lovable content could be consulted.
- Replit connector: available, but requires an explicit `replId` to inspect an app; none was provided, so no Replit content could be consulted.

This report therefore relies on official GitHub documentation + official dataset sites for evidence, plus platform/tool documentation where appropriate. [9]

Repo skeleton and files

Target repo layout

A production-friendly layout that cleanly separates product code, MLOps scripts, infrastructure, monitoring assets, and documentation:

```
.
├── README.md
├── pyproject.toml
├── LICENSE
├── NOTICE
├── SECURITY.md
├── CONTRIBUTING.md
├── CODE_OF_CONDUCT.md
├── CHANGELOG.md
├── .gitignore
├── .dockerignore
├── .pre-commit-config.yaml
├── .editorconfig
├── Makefile
├── docker/
│   ├── Dockerfile
│   └── entrypoint.sh
├── configs/
│   ├── service.yaml
│   ├── logging.yaml
│   ├── model_registry.json
│   └── prometheus/
│       └── prometheus.yml
├── data/
│   ├── README.md
│   ├── metadata/
│   │   ├── datasets/
│   │   │   ├── openimages_v5_masks.md
│   │   │   ├── coco_2017_instances.md
│   │   │   ├── voc2012_segmentation.md
│   │   │   ├── davis2017.md
│   │   │   └── youtube_vos.md
│   │   └── model_cards/
│   │       ├── sam.md
│   │       ├── sam2.md
│   │       └── mobilesam.md
│   └── attribution/
│       ├── DATA_SOURCES.md
│       └── THIRD_PARTY_NOTICES.md
├── scripts/
│   ├── datasets/
│   │   ├── download_openimages_v5.sh
```

```
│   │   ├── download_coco_2017.sh
│   │   ├── download_voc2012.sh
│   │   ├── download_davis2017.sh
│   │   └── download_youtube_vos.sh
│   ├── verify_checksums.py
│   ├── export_onnx.py
│   └── package_release_assets.sh
├── src/
│   └── son_segserve/
│       ├── __init__.py
│       ├── api/
│       │   ├── app.py
│       │   ├── routes.py
│       │   └── schemas.py
│       ├── cli/
│       │   └── main.py
│       ├── core/
│       │   ├── config.py
│       │   ├── logging.py
│       │   └── security.py
│       ├── model/
│       │   ├── loader.py
│       │   ├── registry.py
│       │   ├── prompts.py
│       │   └── postprocess.py
│       ├── metrics/
│       │   ├── prometheus.py
│       │   └── drift.py
│       └── utils/
│           ├── hashing.py
│           └── io.py
├── monitoring/
│   ├── grafana/
│   │   └── dashboard.json
│   └── alerts/
│       └── rules.yml
├── tests/
│   ├── unit/
│   ├── integration/
│   ├── regression/
│   └── synthetic/
├── docs/
│   ├── index.md
│   ├── architecture.md
│   ├── api.md
│   ├── datasets.md
│   ├── models.md
│   ├── contributing.md
│   └── runbooks.md
└── .github/
```

```
    ├── workflows/
    │   ├── ci.yml
    │   ├── docker.yml
    │   ├── docs.yml
    │   └── release.yml
    ├── ISSUE_TEMPLATE/
    │   ├── bug.yml
    │   └── feature_request.yml
    ├── PULL_REQUEST_TEMPLATE.md
    ├── CODEOWNERS
    └── dependabot.yml
```

Design rationale:
- Keep *data* as metadata + scripts only; this supports public redistribution while remaining legally conservative and GitHub-friendly (file size limits, recommended repo size). [10]
- Keep *infra* "portable": Docker + optional compose + sample Prometheus/Grafana configs. Prometheus scrape config patterns are standard. [11]

File-to-purpose mapping table

File / directory	Purpose in a public production repo	Notes / acceptance criteria
README.md	Product definition, quickstart, demo, governance entry points	Must contain "no datasets/weights in git" note
pyproject.toml	Single source of truth for packaging, dependencies, scripts	Supports reproducible local + CI installs
LICENSE	Repo's code licence	Use a standard licence; GitHub recommends adding one for OSS repos [12]
NOTICE	Attribution + legal notices (esp. Apache-style projects)	Include third-party notices + dataset attribution pointers
SECURITY.md	Vulnerability reporting policy	GitHub supports SECURITY.md in root/docs/.github [13]
CONTRIBUTING.md	Contributor workflow and expectations	GitHub recognises standard locations [14]
CODE_OF_CONDUCT.md	Community standards and enforcement	GitHub supports templates and detection [15]
.github/workflows/*.yml	CI/CD automation, pinned and least privilege	Pin SHAs; minimal permissions [16]
.github/dependabot.yml	Dependency update automation	Official recommended approach [17]

File / directory	Purpose in a public production repo	Notes / acceptance criteria
data/metadata/datasets/*.md	Dataset cards for transparency and compliance	Include licence, splits, checksums, provenance notes
scripts/datasets/*	Repeatable dataset download + verification	Never commit dataset binaries
configs/model_registry.json	Machine-readable registry of model artefacts and checksums	Enables download-on-first-run verification
monitoring/*	Dashboards + scrape configs + alerts stub	Grafana dashboards are JSON models [18]

Minimal content templates/snippets

Below are intentionally short templates you can paste in and expand.

README.md (minimal but "public demo ready")

```
# SON SegServe — Promptable Image Segmentation Service

A production-style reference implementation of a promptable segmentation
service:
- REST API (containerised)
- CLI client (thin wrapper over the API)
- Model artefacts fetched at runtime + checksum-verified (no weights
committed)

## Quickstart (Docker)
```bash
docker build -f docker/Dockerfile -t son-segserve:dev .
docker run --rm -p 8080:8080 -e SON_MODEL_ID=sam2_tiny son-segserve:dev
```

Quickstart (CLI)

```
pip install -e ".[dev]"
son-segserve ping --api http://localhost:8080
son-segserve segment --api http://localhost:8080 --image ./demo.jpg --prompt
box:10,10,200,200
```

Repo principles

- Do not commit datasets or model weights to git.
- Use `scripts/datasets/*` to download open datasets locally.
- Use GitHub Releases / OCI artefacts for model checkpoints + checksums.
```

Docs

- Architecture: docs/architecture.md
- API: docs/api.md
- Datasets & licensing: docs/datasets.md
- Contributing: CONTRIBUTING.md
- Security: SECURITY.md

````
#### pyproject.toml (minimal working example)

```toml
[project]
name = "son-segserve"
version = "0.1.0"
description = "Promptable image segmentation service (REST + CLI)"
requires-python = ">=3.10"
readme = "README.md"
license = {file = "LICENSE"}
authors = [{name = "SON Contributors"}]
dependencies = [
  "fastapi>=0.110",
  "uvicorn[standard]>=0.27",
  "pydantic>=2.6",
  "httpx>=0.27",
  "numpy>=1.26",
  "pillow>=10.0",
  "prometheus-client>=0.20",
  # torch / onnxruntime intentionally not pinned here in template:
  # prefer extra groups per backend/hardware to keep default install
light
]

[project.optional-dependencies]
dev = [
  "pytest>=8.0",
  "pytest-cov>=5.0",
  "ruff>=0.4",
  "mypy>=1.8",
  "pre-commit>=4.0",
]
cpu = ["onnxruntime>=1.17"]
gpu = ["onnxruntime-gpu>=1.17"]

[project.scripts]
son-segserve = "son_segserve.cli.main:app"

[tool.ruff]
line-length = 100

[tool.pytest.ini_options]
````
````

```
addopts = "-q"
testpaths = ["tests"]
```

.gitignore (public-ML-safe defaults)

```
# Python
__pycache__/
*.pyc
.venv/
.pytest_cache/
.mypy_cache/
.ruff_cache/

# Local data / checkpoints (never commit)
data/raw/
data/processed/
data/cache/
models/
*.pt
*.pth
*.onnx
*.ckpt

# OS/IDE
.DS_Store
.idea/
.vscode/

# Build outputs
dist/
build/
*.egg-info/
```

.dockerignore (keep build contexts small)

Docker emphasises `.dockerignore` to exclude unnecessary files and reduce build context. [19]

```
.git
.venv
__pycache__
*.pyc
data/raw
data/processed
models
dist
build
```

```
tests
docs
```

.pre-commit-config.yaml (minimal)

pre-commit's recommended approach is to add a `.pre-commit-config.yaml`, then install hooks locally. [20]

```
repos:
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v5.0.0
    hooks:
      - id: end-of-file-fixer
      - id: trailing-whitespace
      - id: check-yaml

  - repo: https://github.com/astral-sh/ruff-pre-commit
    rev: v0.6.4
    hooks:
      - id: ruff
      - id: ruff-format
```

SECURITY.md (minimal)

GitHub explicitly supports adding SECURITY.md with reporting instructions. [13]

```
# Security Policy

## Supported Versions
Only the latest `main` branch and the latest tagged release are supported.

## Reporting a Vulnerability
Please do NOT open a public issue.

Email: security@example.org
Include:
- version/tag/commit SHA
- reproduction steps
- impact assessment (if known)

We aim to acknowledge reports within 72 hours.
```

CONTRIBUTING.md (minimal)

GitHub shows CONTRIBUTING can live in `.github/`, repo root, or `docs/`, with precedence rules. [14]

````
# Contributing

## Development setup
```bash
python -m venv .venv && source .venv/bin/activate
pip install -e ".[dev]"
pre-commit install
pytest
````

Pull request process

- One feature/fix per PR
- Add/adjust tests
- Update docs if behaviour changes
- Ensure `ruff`, `mypy`, and `pytest` pass locally

Code style

- Run `pre-commit run -a` before pushing

````
#### CODE_OF_CONDUCT.md (minimal stub)

GitHub supports adding a code of conduct (ideally via a template). [15]

```md
# Code of Conduct
We follow the Contributor Covenant (recommended). Be respectful and
inclusive.

Enforcement contact: conduct@example.org
````

LICENSE and NOTICE (how to handle in a public repo)

GitHub encourages adding a licence file for OSS. [12]

- `LICENSE`: include the full text for the chosen licence (e.g., Apache-2.0 or MIT).
- `NOTICE`: list third-party notices and required attributions (especially if you choose Apache-2.0 style). Minimal example:

```
SON SegServe
Copyright (c) 2026 SON Contributors

This distribution includes third-party components; see data/attribution/
THIRD_PARTY_NOTICES.md.
Dataset and model licences are documented in data/metadata and must be
reviewed before commercial use.
```

CI/CD workflows

This section gives **YAML outlines** (you will adapt exact Python versions, cache keys, backends). They reflect three non-negotiables for "SON GitHub":

- Least-privilege workflow permissions (job-level `permissions:`).
- Pin Actions to commit SHAs (GitHub explicitly recommends pinning). [16]
- Do not upload raw datasets; only upload build artefacts and small test fixtures.

CI workflow (lint + unit + integration)

```yaml
name: CI

on:
  pull_request:
  push:
    branches: [main]

permissions:
  contents: read

jobs:
  test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python: ["3.10", "3.11"]
    steps:
      - name: Checkout
        uses: actions/checkout@v5

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: ${{ matrix.python }}

      - name: Install
        run: |
          python -m pip install -U pip
          pip install -e ".[dev]"

      - name: Lint
        run: |
          ruff check .
          ruff format --check .

      - name: Type-check
        run: mypy src

      - name: Unit tests
```

```
        run: pytest -q --cov=son_segserve --cov-report=term-missing

  integration:
    runs-on: ubuntu-latest
    needs: [test]
    steps:
      - uses: actions/checkout@v5
      - run: docker build -f docker/Dockerfile -t son-segserve:ci .
      - run: |
          docker run -d --name api -p 8080:8080 son-segserve:ci
          python -m pip install -U httpx
          python scripts/ci_smoke.py --base-url http://localhost:8080
```

Docker build + push workflow (GHCR, optional Docker Hub)

GitHub provides official guidance + example permission sets for publishing container images to GHCR (GitHub Packages) and recommends pinning actions to SHAs. [21]

```
name: Docker

on:
  release:
    types: [published]

env:
  REGISTRY: ghcr.io
  IMAGE_NAME: ${{ github.repository }}

jobs:
  build_push:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write
      attestations: write
      id-token: write
    steps:
      - uses: actions/checkout@v5

      # Login to GHCR using GITHUB_TOKEN
      - name: Login GHCR
        uses: docker/login-action@65b78e6e13532edd9afa3aa52ac7964289d1a9c1
        with:
          registry: ${{ env.REGISTRY }}
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      # OPTIONAL: login to Docker Hub
      # - name: Login Docker Hub
      #   uses: docker/login-action@<PINNED_SHA>
```

```
      #   with:
      #     username: ${{ secrets.DOCKER_USERNAME }}
      #     password: ${{ secrets.DOCKER_PASSWORD }}

      - name: Extract metadata
        id: meta
        uses: docker/metadata-action@9ec57ed1fcdbf14dcef7dfbe97b2010124a938b7
        with:
          images: |
            ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}

      - name: Build and push
        id: push
        uses: docker/build-push-
action@f2a1d5e99d037542a71f64918e516c093c6f3fc4
        with:
          context: .
          file: docker/Dockerfile
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}

      - name: Attest provenance (recommended)
        uses: actions/attest-build-provenance@v3
        with:
          subject-name: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}
          subject-digest: ${{ steps.push.outputs.digest }}
          push-to-registry: true
```

Operational note: GHCR has a **10 GB limit per layer** and a **10 minute upload timeout**, so keep images lean (multi-stage build, exclude caches). [22]

Docs deploy workflow (GitHub Pages)

GitHub Pages "custom workflows" require `pages: write` and `id-token: write`, and using the `github-pages` environment is recommended. [23]

```
name: Docs

on:
  push:
    branches: [main]

permissions:
  contents: read
  pages: write
  id-token: write

jobs:
  build:
```

```
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5
      - uses: actions/setup-python@v5
        with:
          python-version: "3.11"
      - run: |
          python -m pip install -U pip
          pip install -e ".[dev]"
          pip install mkdocs mkdocs-material
          mkdocs build

      - name: Upload Pages artifact
        uses: actions/upload-pages-artifact@v3
        with:
          path: site/

  deploy:
    needs: build
    runs-on: ubuntu-latest
    environment:
      name: github-pages
      url: ${{ steps.deployment.outputs.page_url }}
    steps:
      - name: Deploy
        id: deployment
        uses: actions/deploy-pages@v4
```

Release workflow (create release + attach model artefacts as assets)

GitHub Releases support **up to 1000 assets** per release and **2 GiB per asset**, without total size/bandwidth caps. [6]
To upload assets programmatically, GitHub documents the release asset endpoints and requires sending raw binary bodies to the `upload_url`. [24]

Two robust patterns:

1) **Prefer GitHub CLI in Actions** (simple, readable, portable).
`gh release create` supports attaching assets directly. [25]

2) **Fallback to REST** (more boilerplate; still official). [26]

GitHub CLI-based outline:

```
name: Release

on:
  workflow_dispatch:
  push:
```

```
    tags: ["v*.*.*"]

permissions:
  contents: write

jobs:
  cut_release:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5

      - name: Build model artefact bundle
        run: |
          bash scripts/package_release_assets.sh
          ls -lh dist/

      - name: Create release and upload assets
        env:
          GH_TOKEN: ${{ github.token }}
        run: |
          gh release create "${GITHUB_REF_NAME}" \
            --title "${GITHUB_REF_NAME}" \
            --notes "SON SegServe release ${GITHUB_REF_NAME}" \
            dist/*

      # Optionally verify immutability/asset checks as policy
```

Local equivalent for maintainers (runbook snippet):
`gh auth login` is the documented login entry point. [27]

```
gh auth login
git tag v0.1.0 && git push origin v0.1.0
gh release create v0.1.0 dist/*
```

Model and dataset artefact handling

Why you must not commit weights or datasets

- GitHub blocks files over **100 MiB** in normal git pushes and recommends keeping repositories small (ideal < 1 GB; strongly recommended < 5 GB). [28]
- Git LFS exists, but has **plan-dependent per-file limits** (e.g., 2 GB on Free), and still encourages careful storage strategy. [29]

Hence: **ship download scripts + checksums, not binaries.**

Model registry pattern (`configs/model_registry.json`)

A minimal schema (extend as needed):

```
{
  "schema_version": 1,
  "models": [
    {
      "model_id": "sam2_tiny",
      "family": "sam2",
      "backend": "onnxruntime",
      "format": "onnx",
      "artifact": {
        "uri": "https://github.com/<OWNER>/<REPO>/releases/download/v0.1.0/
sam2_tiny.onnx",
        "sha256": "REPLACE_WITH_SHA256",
        "size_bytes": 123456789
      },
      "license": "requires_review",
      "notes": "Downloaded on first run; cached under XDG cache dir."
    }
  ]
}
```

Download-on-first-run, cache directory, checksum verification

Recommended cache rule (portable):

- Default: `${XDG_CACHE_HOME:-~/.cache}/son-segserve/models/`
- Override via: `SON_CACHE_DIR=/path`

Checksum verification snippet (Python):

```
import hashlib
from pathlib import Path

def sha256_file(path: Path, chunk_size: int = 1024 * 1024) -> str:
    h = hashlib.sha256()
    with path.open("rb") as f:
        for chunk in iter(lambda: f.read(chunk_size), b""):
            h.update(chunk)
    return h.hexdigest()

def verify_sha256(path: Path, expected_hex: str) -> None:
    actual = sha256_file(path)
    if actual.lower() != expected_hex.lower():
        raise ValueError(f"SHA256 mismatch for {path.name}: {actual} !=
{expected_hex}")
```

Principle: **fail closed** (do not load mismatched artefacts).

Artefact distribution options table

Option	Pros	Cons / limits	Best fit for SON GitHub
GitHub Releases	Simple UX; integrates with tags; supports large binaries; no total bandwidth/size limits; 2 GiB per asset; up to 1000 assets per release [6]	Asset management is per-release; you must maintain checksums; 2 GiB per file cap [30]	**Default** for model weights + ONNX exports
GHCR (OCI artefacts / containers)	Good for containerised distribution; integrates with Actions publishing guides [31]	Per-layer 10 GB limit, 10 min upload timeout; requires registry auth and careful layering [32]	Good for shipping the service image; optional for "model-as-OCI"
External object storage (S3/GCS/etc.)	Scales to very large artefacts; independent lifecycle	Requires account + credentials + cost + policy; increases onboarding complexity	Optional for enterprise forks; avoid in baseline

Dataset handling rules and licence flags

Introduce a **dataset compliance gate**: every dataset script must write a `data/processed/<dataset>/MANIFEST.json` containing:

- download URLs
- file checksums (where feasible)
- licence field (`ok_for_demo`, `noncommercial`, `requires_per_item_verification`, etc.)
- citation guidance

Examples of "licence caveats" you must surface in dataset cards:

- Open Images: annotations are CC BY 4.0; images are listed as CC BY 2.0, but the publishers disclaim warranties and advise verifying each image licence. [33]
- YouTube-VOS: annotations under CC BY 4.0, but the dataset is released for **non-commercial research only**—this is generally incompatible with a broadly redistributable public demo intended for mixed use. Mark as "requires review / likely exclude from default demo." [34]
- VOC2012: images sourced from Flickr; use must respect Flickr terms of use (licence is not uniformly permissive). This is a red flag for "public demo assets committed to repo." [35]
- DAVIS: train+val is 90 sequences; evaluation uses region J and boundary F; good for video-seg evaluation but not "tiny quickstart." [36]
- COCO: large-scale; instance segmentation masks; but using subsets for demos is recommended to keep download weight manageable. [37]

Conservative recommendation for SON GitHub demo defaults:

- Default demo dataset: **Open Images masks** (because mask annotations are explicit and large-scale) **but** with prominent "verify image licences" warning. [38]
- Default demo fixtures committed to repo: **synthetic images you generate yourself** (no third-party licence risk).

- Keep YouTube-VOS and VOC off by default in scripts unless the user explicitly accepts their constraints. [39]

Governance and repo lifecycle

Mermaid flowchart for repo lifecycle

```
flowchart LR
  dev[Developer workstation] --> pr[Pull Request]
  pr --> ci[CI: lint + tests + security checks]
  ci -->|pass| merge[Merge to main]
  merge --> build[Build: container + docs]
  build --> release[Release: tag + GH Release assets]
  release --> deploy[Deploy: VM / container runtime]
  deploy --> monitor[Monitor: metrics + logs + drift]
  monitor --> dev
```

Branch protection rules (recommended)

Use GitHub protected branches (or rulesets) to enforce:

- Require PR reviews before merging
- Require status checks before merging
- Require conversation resolution before merging
- Require signed commits (optional but recommended for high-integrity repos)
- Require linear history (optional; helps traceability)

These are standard configurable settings in GitHub's protected branch features. [40]

Also recommended:

- Add `CODEOWNERS` and enable "Require review from Code Owners" to ensure sensitive areas (security, CI, release scripts) are reviewed. [41]

Suggested branch naming strategy

- `main` (protected)
- `feature/<short-scope>`
- `fix/<issue-id>-<short-scope>`
- `release/<version>` (optional, if you gate releases separately)

Recommended GitHub settings to enable

Security posture features that are explicitly supported for public repos:

- **Secret scanning** can be enabled for free public repositories you own. [42]
- **Dependabot version updates** via `.github/dependabot.yml`. [17]
- **CodeQL code scanning default setup** is eligible for public repos with Actions enabled; scans on pushes/PRs and on a schedule. [43]

Tests, monitoring, security, reproducibility, and runbooks

Test suite inventory you should include

Keep tests small, deterministic, and licence-safe:

- Unit tests: pure functions (hashing, config parsing, prompt parsing, mask post-process).
- Integration tests: start container, hit `/healthz`, `/segment`, `/metrics`.
- Model regression ("golden set"):
- Use generated synthetic images + fixed prompts; store expected *summary statistics* (mask area, connected components count) instead of raw masks to avoid brittle exact pixel equality.
- Track a hash of the model artefact + inference backend version in the regression output.
- Data validation tests: validate dataset manifests and expected directory layouts after download (but do not run full dataset downloads in CI).
- Synthetic tests: fuzz prompts (invalid box coords, empty points, huge images) to harden request validation and avoid crashes.

Local run commands (developer runbook):

```
pip install -e ".[dev]"
pre-commit run -a
pytest
docker build -f docker/Dockerfile -t son-segserve:dev .
docker run --rm -p 8080:8080 son-segserve:dev
```

Monitoring and observability hooks

Metrics endpoint spec

Expose `GET /metrics` in Prometheus text format with `Content-Type: text/plain; version=0.0.4`. [44]

Metrics to collect (minimum viable):

- HTTP request count + status
- Request latency histogram (p50/p95 derived in Prometheus/Grafana)
- Inference time histogram
- Memory RSS gauge (process metric)
- Mask count per request (gauge/hist)
- Predicted quality proxy (e.g., `predicted_iou` distribution)

Prometheus config example (scrape your service):

Prometheus YAML structures (`scrape_configs`) are documented and standard. [45]

```
global:
  scrape_interval: 15s

scrape_configs:
```

```
  - job_name: son-segserve
    metrics_path: /metrics
    static_configs:
      - targets: ["host.docker.internal:8080"]
```

Sample Grafana dashboard JSON stub

Grafana dashboards are JSON objects; Grafana documents the JSON model. [46]

```
{
  "id": null,
  "uid": "son-segserve",
  "title": "SON SegServe",
  "tags": ["segmentation", "son"],
  "timezone": "browser",
  "schemaVersion": 41,
  "panels": [
    {
      "type": "timeseries",
      "title": "HTTP latency (p95)",
      "targets": [
        { "expr": "histogram_quantile(0.95,
sum(rate(http_request_duration_seconds_bucket[5m])) by (le))" }
      ],
      "gridPos": { "h": 8, "w": 24, "x": 0, "y": 0 }
    }
  ]
}
```

Security and licensing checks

Licence auditing workflow (conservative)

1) **Model licence**: verify upstream terms (code + checkpoints) and write a "model card" under `data/metadata/model_cards/`. If any ambiguity exists, set `license: requires_review` in `model_registry.json` and do not include weights in default releases.
2) **Dataset licence**: record dataset homepage, licence text, and restrictions in dataset card.
3) **Repo licence**: choose a clear repo licence (Apache-2.0 or MIT are common); GitHub supports licence detection when using standard filenames. [12]
4) **Enforce a "no large files" policy**: include CI check that fails if a PR adds `*.pt`, `*.onnx`, or files > a threshold. This aligns with GitHub's large file limits and prevents accidental commits. [47]

Licence ambiguity flags you should raise in SON GitHub dataset cards:

- YouTube-VOS: **non-commercial research only** → treat as **not suitable** for a broadly reusable public demo by default. [48]
- Open Images: licence verification required per image (explicit disclaimer) → acceptable for local evaluation, but do not commit images into your repo; provide download scripts only. [33]
- VOC: Flickr terms apply → same: scripts only, no committed images. [35]

Reproducibility steps (what to implement)

- Record seeds and deterministic settings in `configs/service.yaml` and print them at startup.
- Add a runtime `--deterministic` flag: enables deterministic algorithms where possible and documents potential speed impacts.
- Pin container base images and keep Dockerfiles multi-stage as per Docker best practices. [49]
- Pin GitHub Actions to commit SHAs. [16]
- For releases: include artefact checksums alongside binaries as separate `*.sha256` assets; GitHub release assets are supported and downloadable. [50]

Dockerfile (recommended baseline)

Docker best practices emphasise multi-stage builds and `.dockerignore` use; multi-stage builds are a first-class Docker feature. [51]

```
# syntax=docker/dockerfile:1

FROM python:3.11-slim AS build
WORKDIR /app

# System deps only if needed (keep minimal)
RUN python -m pip install -U pip

COPY pyproject.toml README.md LICENSE /app/
COPY src/ /app/src/

# Install as editable into a venv-like layer
RUN pip install --no-cache-dir -e ".[cpu]"

FROM python:3.11-slim AS runtime
WORKDIR /app

# Non-root user is recommended for runtime images
RUN useradd -m -u 10001 appuser
USER appuser

COPY --from=build /usr/local /usr/local
COPY --from=build /app/src /app/src
COPY configs/ /app/configs
COPY docker/entrypoint.sh /app/entrypoint.sh

EXPOSE 8080
ENV SON_CONFIG=/app/configs/service.yaml
ENTRYPOINT ["/app/entrypoint.sh"]
```

`docker/entrypoint.sh`:

```
#!/usr/bin/env bash
set -euo pipefail
exec python -m son_segserve.api.app --host 0.0.0.0 --port 8080
```

REST API spec (minimum viable)

Endpoints (versioned):

- `GET /healthz` → service health
- `GET /readyz` → readiness (model loaded, cache ok)
- `GET /metrics` → Prometheus metrics (text format) [44]
- `POST /v1/segment` → run segmentation from image + prompt
- `POST /v1/segment/video` → optional: short video segmentation with temporal smoothing (advanced)

`POST /v1/segment` request JSON (URLs or base64 are typical; avoid multipart in first public demo unless needed):

```
{
  "image": {
    "content_type": "image/jpeg",
    "base64": "..."
  },
  "prompt": {
    "type": "box",
    "box_xyxy": [10, 20, 200, 220]
  },
  "options": {
    "return_format": "rle",
    "max_masks": 3,
    "stabilize": false
  }
}
```

Response JSON:

```
{
  "model_id": "sam2_tiny",
  "masks": [
    {
      "mask_id": "0",
      "encoding": "rle",
      "rle": "....",
      "predicted_iou": 0.87,
      "area_px": 12345
    }
  ],
  "timing_ms": {
```

```
    "preprocess": 4.1,
    "inference": 31.7,
    "postprocess": 7.9,
    "total": 43.7
  }
}
```

CLI commands (thin client)

CLI should support:

```
son-segserve ping --api http://localhost:8080
son-segserve segment --api http://localhost:8080 --image ./x.jpg --prompt
"box:10,20,200,220" --out ./mask.json
son-segserve bench --api http://localhost:8080 --images ./fixtures --prompt
"point:120,88,fg"
```

PR checklist table

Category	Checklist item (concrete)	How reviewer verifies
Scope	PR is single-purpose and titled clearly	Title + diff inspection
Tests	New/changed behaviour has tests	`pytest` passes locally/CI
CI hygiene	No new workflow secrets added without justification	`.github/workflows` diff
Artefact safety	No weights/datasets/binaries committed	repo diff + CI “large file guard”
Licensing	Dataset/model licence implications documented (if touched)	dataset/model card updated
Reproducibility	Seeds/config updated when changing model behaviour	`configs/service.yaml` + logs
Observability	Metrics/logging updated for new paths	`/metrics` samples + logs
Docs	README/docs updated if UX changes	docs diff + preview build

Assumptions

Unspecified items treated as assumptions:

- Cloud provider is unspecified → recommend “portable baseline” with a single container on a generic VM (or local).
- Hardware is unspecified → assume **GPU optional**; provide CPU ONNXRuntime path by default, GPU as an extra.
- Exact model size is unspecified → assume multiple model IDs (tiny/base/large) via `model_registry.json`.

- Licence clarity for some model checkpoints is unspecified / constrained by the instruction not to consult upstream GitHub repos → therefore model licences are marked **requires review** unless verified by an official artefact you explicitly approve.
- Public repo requirement implies: do not commit training data, do not commit weights unless you have explicit redistribution rights, and do not include any non-commercial-only datasets in the default demo path. [52]

Sources

Primary/official documentation and dataset sources used for evidence:

- GitHub large files, repo size guidance, and releases as a distribution method. [28]
- GitHub Releases limits (1000 assets/release, 2 GiB/asset, no total size/bandwidth cap). [30]
- GitHub Actions: publishing Docker images, required permissions, and recommendation to pin Actions to SHAs. [53]
- GitHub Pages custom workflow permissions (`pages:write`, `id-token:write`) and deployment patterns. [23]
- GitHub Container Registry layer/time limits. [32]
- GitHub security features: secret scanning availability for public repos. [42]
- Dependabot configuration via `.github/dependabot.yml`. [17]
- CodeQL default setup eligibility and behaviour. [43]
- Branch protection settings and options. [54]
- CODEOWNERS usage with branch protection. [41]
- Prometheus exposition format (`text/plain; version=0.0.4`) and scrape configuration structure. [55]
- Grafana dashboard JSON model reference. [46]
- Docker build best practices (multi-stage, `.dockerignore`, smaller images). [51]
- pre-commit quickstart and configuration expectations. [20]
- Open Images dataset facts/figures and licence notes (CC BY 4.0 annotations; CC BY 2.0 images listed; verify per image). [56]
- YouTube-VOS terms (non-commercial research only; annotations under CC BY 4.0) and dataset statistics. [34]
- DAVIS dataset splits and evaluation metrics (J & F). [36]
- COCO overview and scale (COCO as detection/segmentation dataset; overall scale). [37]
- VOC2012 database rights note (Flickr terms). [35]

Primary links requested for models/datasets (provided as URLs only; not used as evidence here due to the constraint to avoid other GitHub repos):

```
Models (primary links to verify licences/checkpoints):
- SAM: https://segment-anything.com/
- SAM 2: https://ai.meta.com/sam2/
- MobileSAM: https://arxiv.org/abs/2306.14289  (paper) and upstream repo
(verify separately)

Datasets:
- COCO: https://cocodataset.org/
- PASCAL VOC: http://host.robots.ox.ac.uk/pascal/VOC/
- Open Images: https://storage.googleapis.com/openimages/web/index.html
```

```
- DAVIS: https://davischallenge.org/
- YouTube-VOS: https://youtube-vos.org/
```

---

1 11 45 Configuration | Prometheus
https://prometheus.io/docs/prometheus/3.4/configuration/configuration/?utm_source=chatgpt.com

2 40 54 https://docs.github.com/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/about-protected-branches
https://docs.github.com/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/about-protected-branches

3 24 REST API endpoints for release assets - GitHub Docs
https://docs.github.com/en/rest/releases/assets?utm_source=chatgpt.com

4 5 9 10 28 47 https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github
https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github

6 30 50 https://docs.github.com/free-pro-team%40latest/github/administering-a-repository/about-releases
https://docs.github.com/free-pro-team%40latest/github/administering-a-repository/about-releases

7 35 https://www.robots.ox.ac.uk/~vgg/projects/pascal/VOC/voc2012/index.html
https://www.robots.ox.ac.uk/~vgg/projects/pascal/VOC/voc2012/index.html

8 16 21 31 53 https://docs.github.com/en/actions/tutorials/publish-packages/publish-docker-images
https://docs.github.com/en/actions/tutorials/publish-packages/publish-docker-images

12 https://docs.github.com/articles/adding-a-license-to-a-repository
https://docs.github.com/articles/adding-a-license-to-a-repository

13 https://docs.github.com/articles/adding-a-security-policy-to-your-repository
https://docs.github.com/articles/adding-a-security-policy-to-your-repository

14 https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/setting-guidelines-for-repository-contributors?trk=public_post_comment-text
https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/setting-guidelines-for-repository-contributors?trk=public_post_comment-text

15 https://docs.github.com/articles/adding-a-code-of-conduct-to-your-project
https://docs.github.com/articles/adding-a-code-of-conduct-to-your-project

17 https://docs.github.com/en/code-security/dependabot/dependabot-version-updates/configuring-dependabot-version-updates
https://docs.github.com/en/code-security/dependabot/dependabot-version-updates/configuring-dependabot-version-updates

18 46 JSON model | Grafana documentation
https://grafana.com/docs/grafana/latest/reference/dashboard/?utm_source=chatgpt.com

19 Optimize usage | Docker Docs
https://docs.docker.com/offload/optimize/?utm_source=chatgpt.com

20 pre-commit
https://pre-commit.com/?utm_source=chatgpt.com

22 32 https://docs.github.com/en/enterprise-cloud%40latest/packages/working-with-a-github-packages-registry/working-with-the-container-registry
https://docs.github.com/en/enterprise-cloud%40latest/packages/working-with-a-github-packages-registry/working-with-the-container-registry

23 https://docs.github.com/pages/getting-started-with-github-pages/using-custom-workflows-with-github-pages
https://docs.github.com/pages/getting-started-with-github-pages/using-custom-workflows-with-github-pages

25 GitHub CLI | Take GitHub to the command line
https://cli.github.com/manual/gh_release_create?utm_source=chatgpt.com

26 REST API endpoints for releases - GitHub Docs
https://docs.github.com/en/rest/releases/releases?utm_source=chatgpt.com

27 GitHub CLI | Take GitHub to the command line
https://cli.github.com/manual/gh_auth_login?utm_source=chatgpt.com

29 https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-git-large-file-storage
https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-git-large-file-storage

33 38 56 https://storage.googleapis.com/openimages/web/factsfigures_v5.html
https://storage.googleapis.com/openimages/web/factsfigures_v5.html

34 39 48 52 https://youtube-vos.org/dataset/term/
https://youtube-vos.org/dataset/term/

36 https://davischallenge.org/challenge2019/semisupervised.html
https://davischallenge.org/challenge2019/semisupervised.html

37 https://cocodataset.org/dataset/home.htm
https://cocodataset.org/dataset/home.htm

41 https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-code-owners?ref=the-mergify-blog
https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-code-owners?ref=the-mergify-blog

42 https://docs.github.com/en/code-security/secret-scanning/enabling-secret-scanning-features/enabling-secret-scanning-for-your-repository
https://docs.github.com/en/code-security/secret-scanning/enabling-secret-scanning-features/enabling-secret-scanning-for-your-repository

43 https://docs.github.com/en/code-security/how-tos/scan-code-for-vulnerabilities/configure-code-scanning/configuring-default-setup-for-code-scanning
https://docs.github.com/en/code-security/how-tos/scan-code-for-vulnerabilities/configure-code-scanning/configuring-default-setup-for-code-scanning

44 55 Exposition formats | Prometheus
https://prometheus.io/docs/instrumenting/exposition_formats/?utm_source=chatgpt.com

49 51 Best practices | Docker Docs
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/?utm_source=chatgpt.com